Done by-

Name	Roll Number	Gr Number
Aaditya Diwan	323001	17u546
Ankit Bawanthade	323009	17u693
Bhupendra Nagda	323014	17u161
Jayanth Thopil	323019	17u221

Title – Air Pollution Data Visualization and Prediction

Objectives –

1. Do the statistical analysis of the data
2. Replace missing numerical values by its central tendency
3. Find appropriate filler for attributes with categorical values
4. Render the data fit for a Machine Learning model
    a. Label Encode the categorical values.
    b. Apply OneHotEncoder in case there are more than two types of encoded values.
    c. Join the datasets formed to get final dataset
5. Apply Machine Learning model for prediction

Dataset Attributes –

The main Dataset consisted of 13 columns –

1. Stn_code – Numerical value
2. Sampling_Date- String
3. State- Categorical Value
4. Location-Categorical Value

5. Agency- Categorical Value
6. Type- Categorical Value
7. SO2- Numerical value
8. NO2- Numerical value
9. RSPM- Numerical value
10. SPM- Numerical value
11. Location_monitoring_system- Categorical Value
12. Pm2_5- Numerical value
13. Date-String

Dataset Details –

- The main attributes that need attention are –
    - **Date**
        - The date on which particular rec
    - **SO2**
        - So2 is the fundamental cause for acid rain. It causes various diseases in humans too and has adverse effects too. Main causes are Burning fuels, industrial areas, etc.
    - **NO2**
        - Similar to SO2 it has adverse effects. Main contributors are vehicles.
    - **Pm2_5**
        - They are called particulate matter. They have a diameter of less than 2.5 micrometer they are the most dangerous in all. Since very small, can't be seen and can cause various cardiovascular diseases depending on exposure. They are emmited by factories, industries, etc
    - **Rspm**

        - They are called as residual particulate matter or also as pm10 i.e they have diameter of roughly less than 10 micrometer. they are less hazardous than pm2.5 but hazardous nonetheless. Emmited by factories, etc
    - **Spm**
        - Same as rspm, but have smaller size
    - **Type**
        - It tells us about the area. i.e- whether Industrial, Residential Rural or other
    - **Location & State**
        - Tell the location and state of the data recorded.

Preprocessing –

- The data was in structured format. The main problem with the data was missing values. The steps taken to rectify are as follows
    1. Fill the blank spaces with NaN values using numpy for further ease.
    2. If the data is numerical, find the best central tendency and replace the NaN values with it. In our case we took mean as central tendency.
    3. If the data is categorical, replace it with either Backfill or Front fill.
- Since many attributes such as stn_code, sampling_date, agency and location_montoring_system were redundant, drop them.
- The next step was to convert the categorical data into discrete format. The steps taken were-
    1. Use the library LabelEncoder on Categorical attributes like type and location to convert it into discrete values.
    2. Since discrete values may skew our model, we need to convert any former categorical attribute having 2 or more subtypes into separate

columns having binary values. For this, OneHotEncoder library was used.

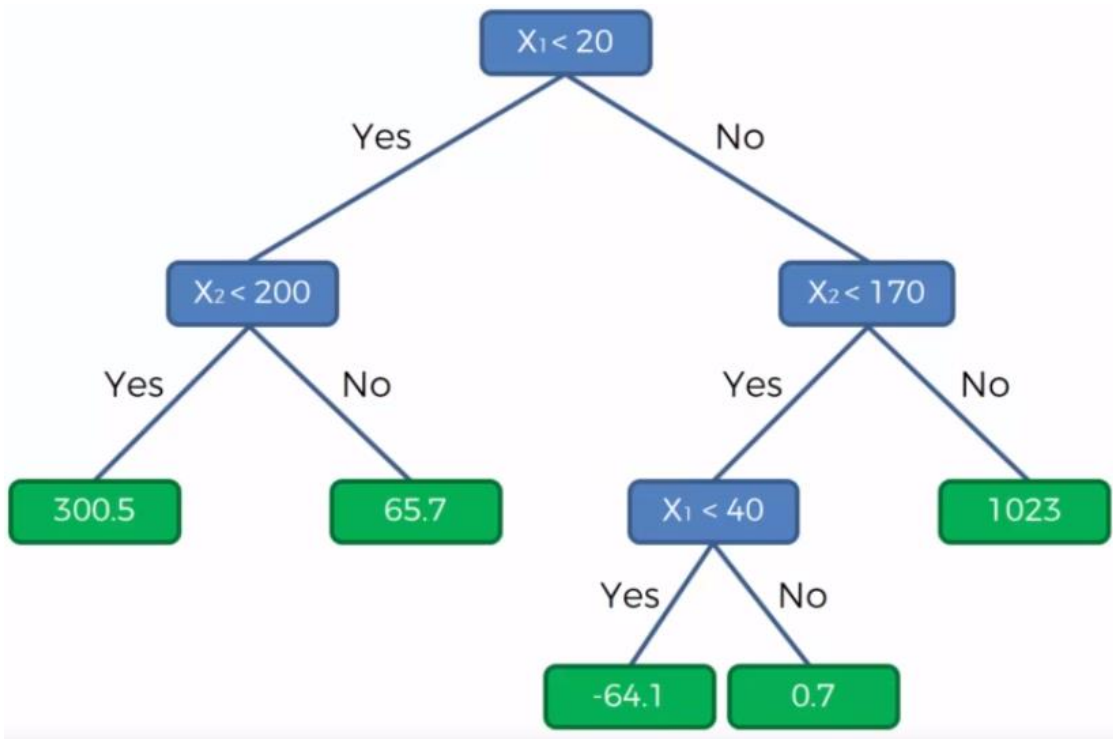
- These dataFrame were merged to get a single dataset having 31 attributes.
- This dataFrame was segregated into 2 different dataFrames X and y. X being the dataFrame used to predict the y dataFrame.

Regressor-

- Decision Tree regressor was used to determine the numerical values.

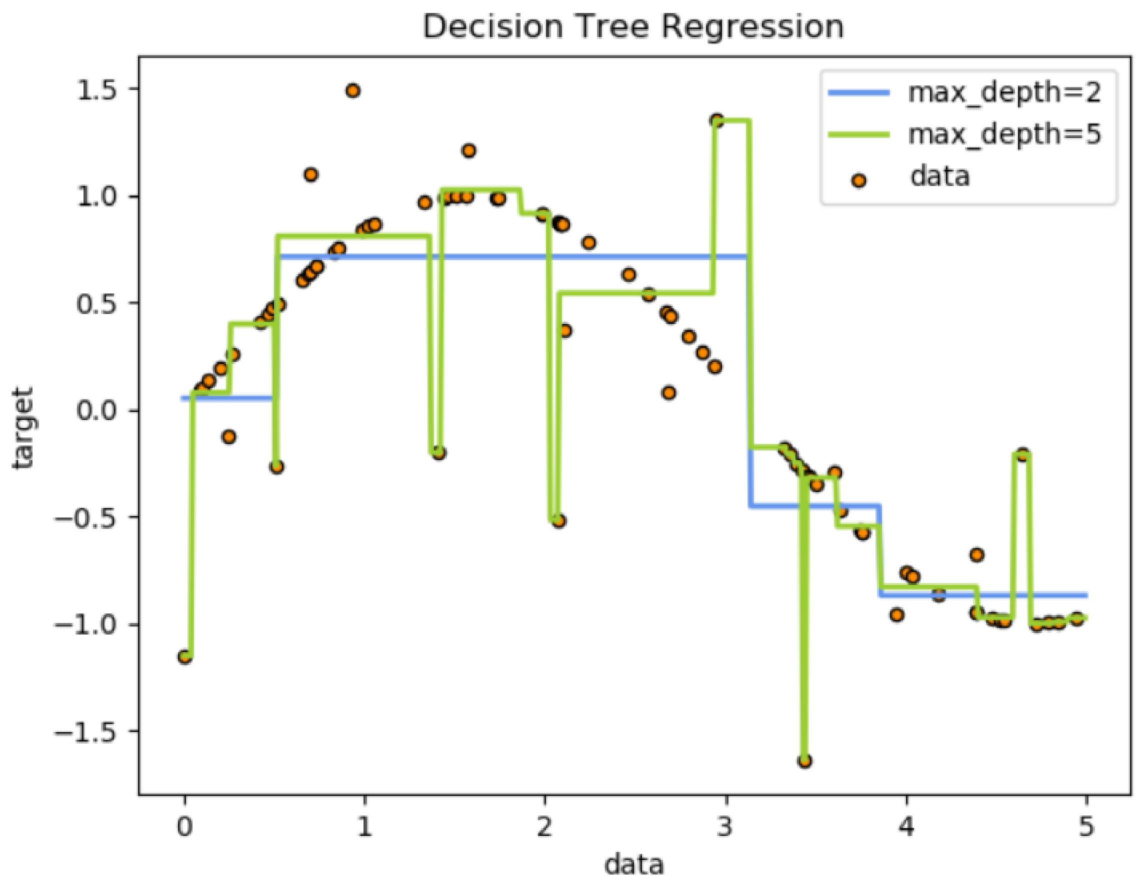
Theory behind Decision Trees-

- Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**.
- The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*.



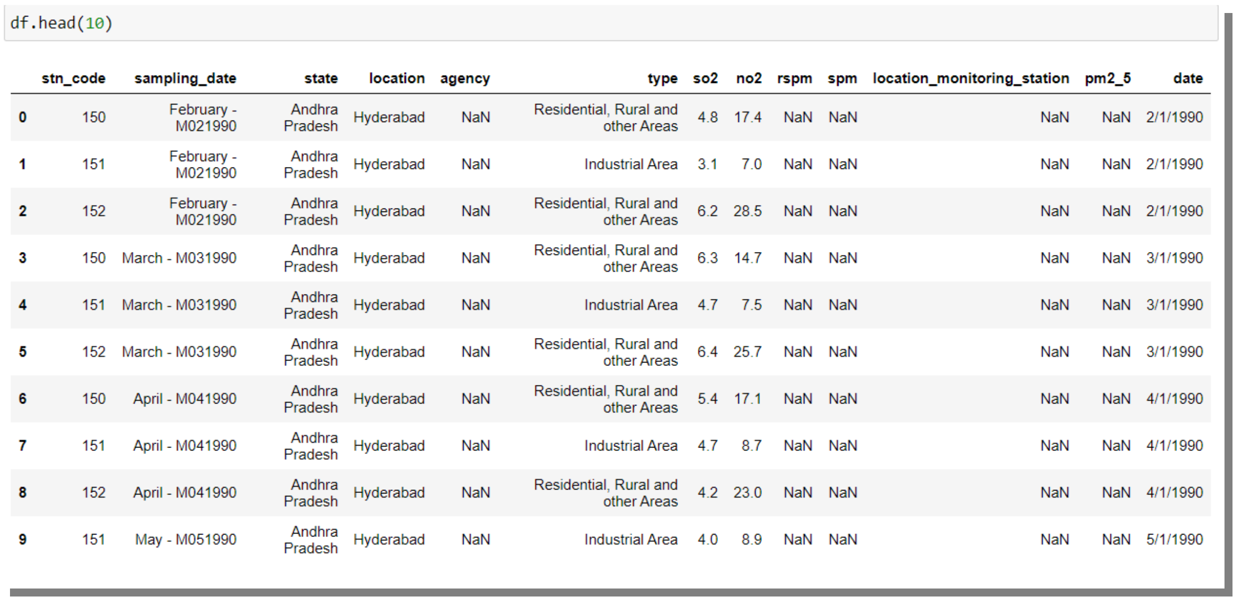
A simple representation of decision tree.

- The decision tree is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.
- We can see that if the maximum depth of the tree (controlled by the max_depth parameter) is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.



Results-

- Data-Interpretation –

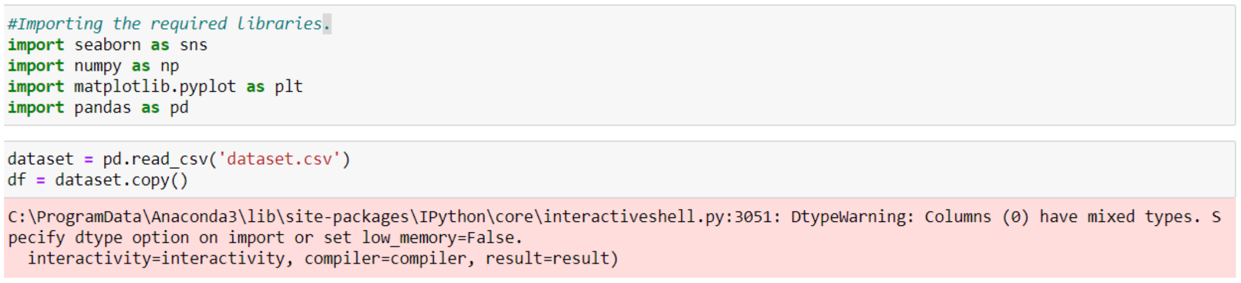


```
df.head(10)
```

	stn_code	sampling_date	state	location	agency	type	so2	no2	rspm	spm	location_monitoring_station	pm2_5	date
0	150	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	4.8	17.4	NaN	NaN	NaN	NaN	2/1/1990
1	151	February - M021990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	3.1	7.0	NaN	NaN	NaN	NaN	2/1/1990
2	152	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.2	28.5	NaN	NaN	NaN	NaN	2/1/1990
3	150	March - M031990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.3	14.7	NaN	NaN	NaN	NaN	3/1/1990
4	151	March - M031990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	4.7	7.5	NaN	NaN	NaN	NaN	3/1/1990
5	152	March - M031990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.4	25.7	NaN	NaN	NaN	NaN	3/1/1990
6	150	April - M041990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	5.4	17.1	NaN	NaN	NaN	NaN	4/1/1990
7	151	April - M041990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	4.7	8.7	NaN	NaN	NaN	NaN	4/1/1990
8	152	April - M041990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	4.2	23.0	NaN	NaN	NaN	NaN	4/1/1990
9	151	May - M051990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	4.0	8.9	NaN	NaN	NaN	NaN	5/1/1990

```
df.info()
df.isnull().sum()
```

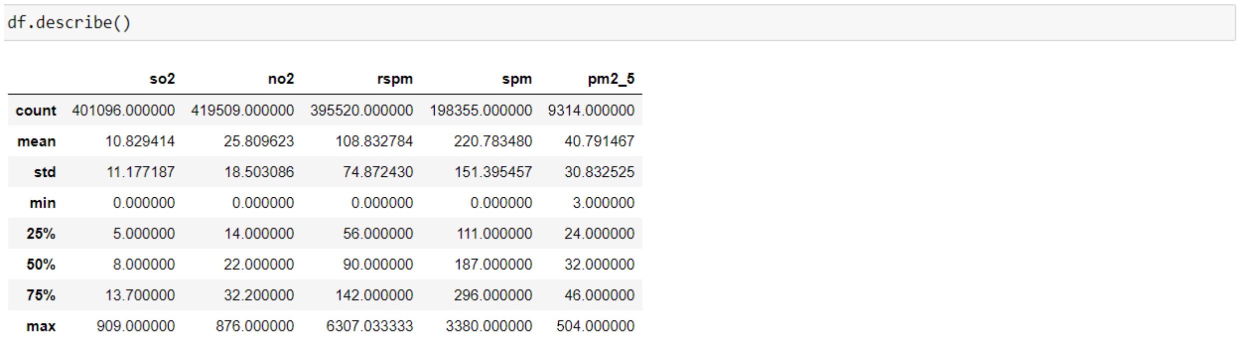
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
Data columns (total 13 columns):
stn_code                       291665 non-null object
sampling_date                  435739 non-null object
state                          435742 non-null object
location                       435739 non-null object
agency                         286261 non-null object
type                           430349 non-null object
so2                            401096 non-null float64
no2                            419509 non-null float64
rspm                           395520 non-null float64
spm                            198355 non-null float64
location_monitoring_station    408251 non-null object
pm2_5                          9314 non-null float64
date                           435735 non-null object
dtypes: float64(5), object(8)
memory usage: 43.2+ MB

stn_code                       144077
sampling_date                       3
state                               0
location                            3
agency                         149481
type                             5393
so2                             34646
no2                             16233
rspm                            40222
spm                            237387
location_monitoring_station     27491
pm2_5                          426428
date                                7
dtype: int64
```



```
#Importing the required libraries.
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

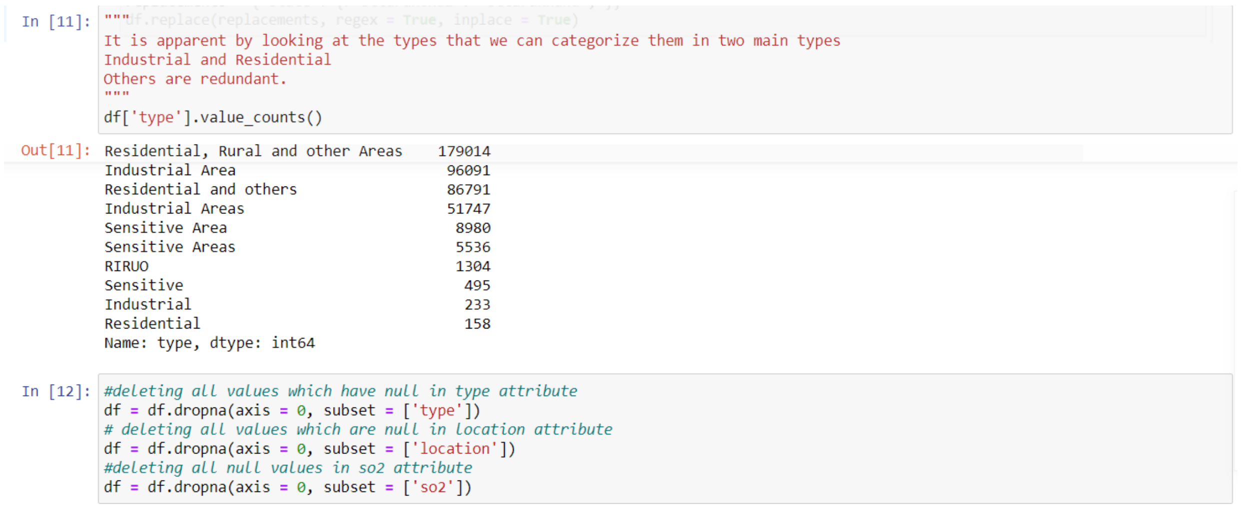
```
dataset = pd.read_csv('dataset.csv')
df = dataset.copy()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3051: DtypeWarning: Columns (0) have mixed types. S
pecify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```



```
df.describe()
```

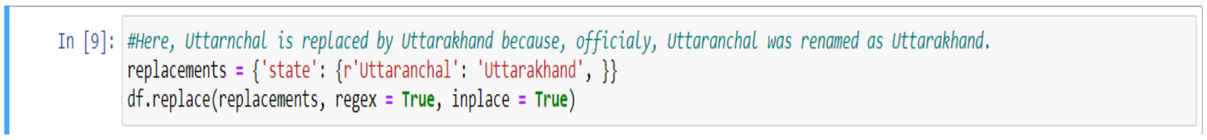
	so2	no2	rspm	spm	pm2_5
count	401096.000000	419509.000000	395520.000000	198355.000000	9314.000000
mean	10.829414	25.809623	108.832784	220.783480	40.791467
std	11.177187	18.503086	74.872430	151.395457	30.832525
min	0.000000	0.000000	0.000000	0.000000	3.000000
25%	5.000000	14.000000	56.000000	111.000000	24.000000
50%	8.000000	22.000000	90.000000	187.000000	32.000000
75%	13.700000	32.200000	142.000000	296.000000	46.000000
max	909.000000	876.000000	6307.033333	3380.000000	504.000000



```
In [11]: """df.replace(replacements, regex = True, inplace = True)
         It is apparent by looking at the types that we can categorize them in two main types
         Industrial and Residential
         Others are redundant.
         """
         df['type'].value_counts()

Out[11]: Residential, Rural and other Areas    179014
         Industrial Area                        96091
         Residential and others                 86791
         Industrial Areas                       51747
         Sensitive Area                          8980
         Sensitive Areas                         5536
         RIRUO                                   1304
         Sensitive                                495
         Industrial                               233
         Residential                              158
         Name: type, dtype: int64
```

```
In [12]: #deleting all values which have null in type attribute
         df = df.dropna(axis = 0, subset = ['type'])
         # deleting all values which are null in location attribute
         df = df.dropna(axis = 0, subset = ['location'])
         #deleting all null values in so2 attribute
         df = df.dropna(axis = 0, subset = ['so2'])
```

- Data Preprocessing –



```
In [9]: #Here, Uttarnchal is replaced by Uttarakhand because, officialy, Uttaranchal was renamed as Uttarakhand.
        replacements = {'state': {r'Uttaranchal': 'Uttarakhand', }}
        df.replace(replacements, regex = True, inplace = True)
```

```
In [14]: del df['agency']
         del df['location_monitoring_station']
         del df['stn_code']
         del df['sampling_date']
```

```
In [15]: df.head()
```

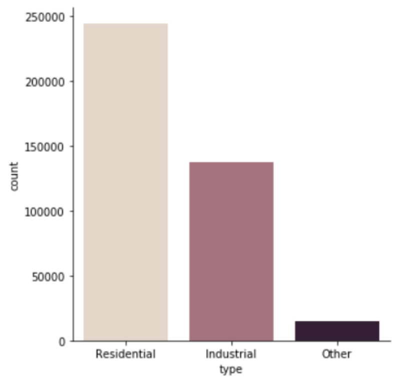
Out[15]:

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	Andhra Pradesh	Hyderabad	Residential, Rural and other Areas	4.8	17.4	NaN	NaN	NaN	2/1/1990
1	Andhra Pradesh	Hyderabad	Industrial Area	3.1	7.0	NaN	NaN	NaN	2/1/1990
2	Andhra Pradesh	Hyderabad	Residential, Rural and other Areas	6.2	28.5	NaN	NaN	NaN	2/1/1990
3	Andhra Pradesh	Hyderabad	Residential, Rural and other Areas	6.3	14.7	NaN	NaN	NaN	3/1/1990
4	Andhra Pradesh	Hyderabad	Industrial Area	4.7	7.5	NaN	NaN	NaN	3/1/1990

```
In [16]: a = list(df['type'])
         for i in range(0, len(df)):
             if str(a[i][0]) == 'R' and a[i][1] == 'e':
                 a[i] = 'Residential'
             elif str(a[i][0]) == 'I':
                 a[i] = 'Industrial'
             else:
                 a[i] = 'Other'

         df['type'] = a
         df['type'].value_counts()

Out[16]: Residential    244017
         Industrial     137420
         Other           14724
         Name: type, dtype: int64
```

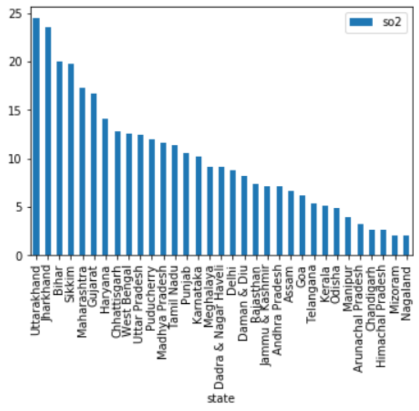
As mentioned above, We can remove the redundant types and get only 2 main

```
In [17]: #how many observations belong to each location
         sns.catplot(x = "type", kind = "count", palette = "ch: 0.25", data = df)

Out[17]: <seaborn.axisgrid.FacetGrid at 0x1c91292d3c8>
```

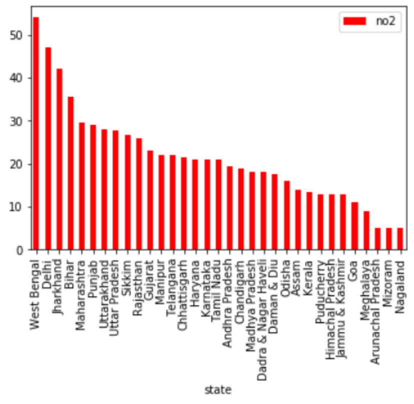


- Data Visualization entire dataset-

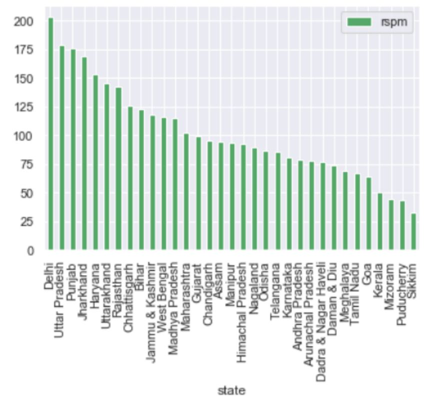
```
In [18]: #bar plot of so2 vs state - desc order
         df[['so2', 'state']].groupby(['state']).mean().sort_values("so2", ascending = False).plot.bar()

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1c924f1b4e0>
```



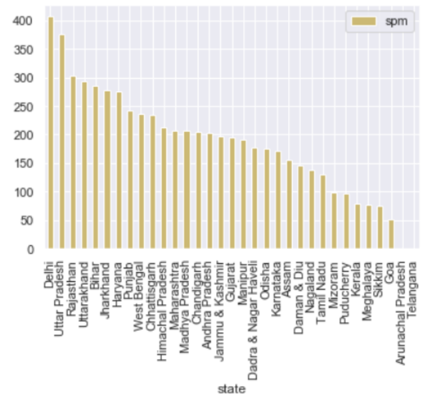
```
In [19]: # bar plot of no2 vs state - desc order
         df[['no2', 'state']].groupby(['state']).median().sort_values("no2", ascending = False).plot.bar(color = 'r')

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1c91293fe10>
```

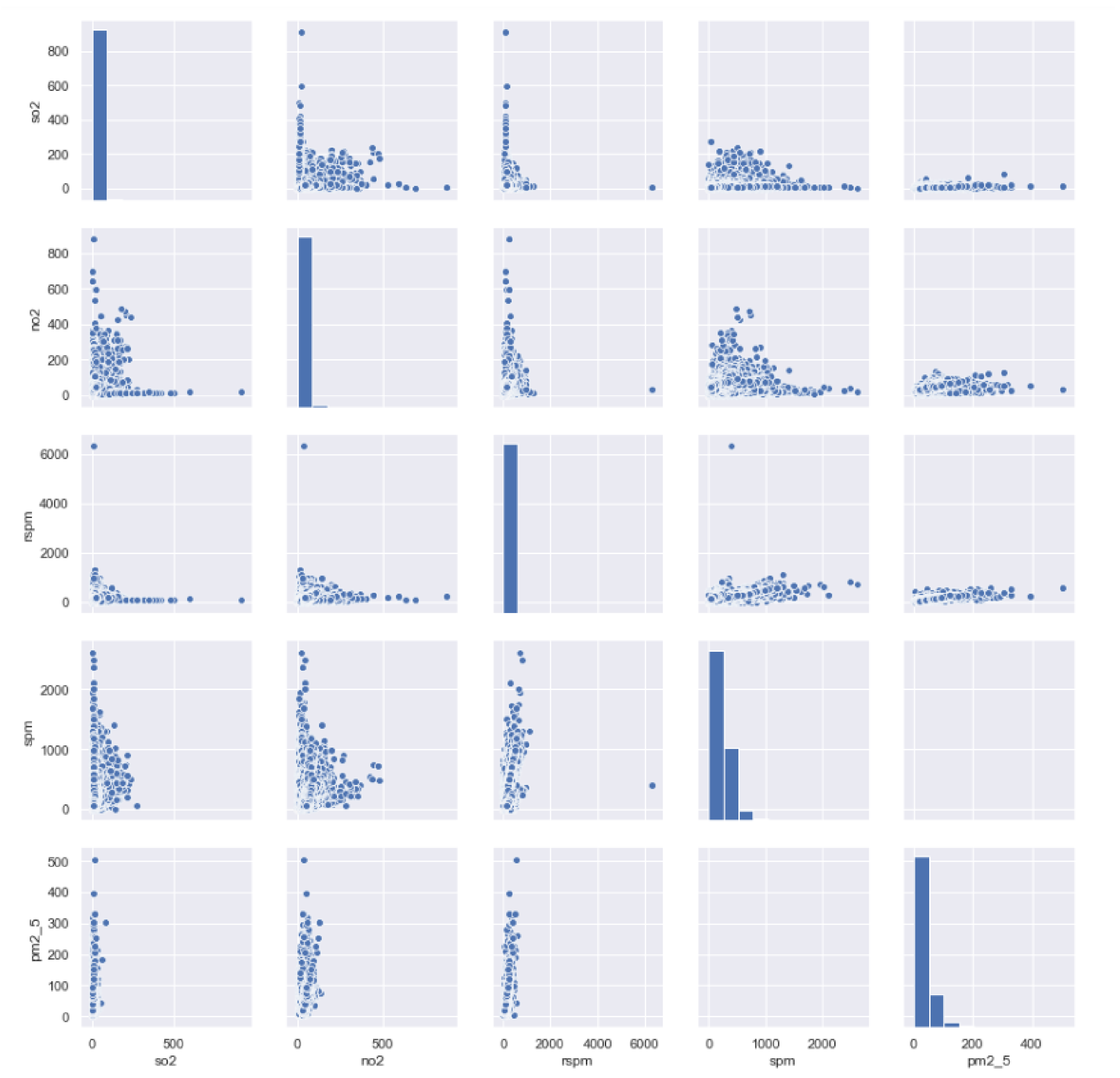


```
In [125]: # rspm = PM10
          df[['rspm', 'state']].groupby(['state']).mean().sort_values("rspm", ascending = False).plot.bar(color = 'g')

Out[125]: <matplotlib.axes._subplots.AxesSubplot at 0x1c91a16ab00>
```



```
In [127]: # spm
          df[['spm', 'state']].groupby(['state']).mean().sort_values("spm", ascending = False).plot.bar(color = 'y')

Out[127]: <matplotlib.axes._subplots.AxesSubplot at 0x1c91f5894a8>
```





Code –

```
#Importing the required libraries.

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

```
dataset = pd.read_csv('dataset.csv')
df = dataset.copy()
df.describe()
df['type'].nunique()
df['type'].unique()

df.info()
df.isnull().sum()
#Here, Uttarnchal is replaced by Uttarakhand because, officialy, Uttaranchal
,→was renamed as Uttarakhand.
replacements = {'state': {r'Uttaranchal': 'Uttarakhand', }}
df.replace(replacements, regex = True, inplace = True)

"""
It is apparent by looking at the types that we can categorize them in two main
,→types
Industrial and Residential
Others are redundant.
"""
df['type'].value_counts()

#deleting all values which have null in type attribute
df = df.dropna(axis = 0, subset = ['type'])
# deleting all values which are null in location attribute
df = df.dropna(axis = 0, subset = ['location'])
#deleting all null values in so2 attribute
df = df.dropna(axis = 0, subset = ['so2'])
```

```
del df['agency']

del df['location_monitoring_station']

del df['stn_code']

del df['sampling_date']


a = list(df['type'])

for i in range(0, len(df)):

if str(a[i][0]) == 'R' and a[i][1] == 'e':

a[i] = 'Residential'

elif str(a[i][0]) == 'I':

a[i] = 'Industrial'

else:

a[i] = 'Other'

df['type'] = a

df['type'].value_counts()


#how many observations belong to each location

sns.catplot(x = "type", kind = "count", palette = "ch: 0.25", data = df)


#bar plot of so2 vs state - desc order

df[['so2', 'state']].groupby(['state']).mean().sort_values("so2", ascending =�

,→False).plot.bar()


# bar plot of no2 vs state - desc order

df[['no2', 'state']].groupby(['state']).median().sort_values("no2", ascending =�

,→False).plot.bar(color = 'r')


# rspm = PM10

df[['rspm', 'state']].groupby(['state']).mean().sort_values("rspm", ascending =�
,→False).plot.bar(color = 'g')

# spm
df[['spm', 'state']].groupby(['state']).mean().sort_values("spm", ascending =␣
,→False).plot.bar(color = 'y')

# pm2_5
df[['pm2_5', 'state']].groupby(['state']).mean().sort_values("pm2_5", ascending␣
,→= False).plot.bar(color = 'r')

#Scatter plots of all columns
sns.set()
cols = ['so2', 'no2', 'rspm', 'spm', 'pm2_5']
sns.pairplot(df[cols], size = 2.5)
plt.show()

corrmat = df.corr()
f, ax = plt.subplots(figsize = (15, 10))
sns.heatmap(corrmat, vmax = 1, annot = True, square = True)

"""
Creating a seperate dataframe for Andhra Pradesh having all the properties
"""
df_andhra = df.iloc[0:25086,:]

df_andhra.info()

"""
Dropping pm2_5 as it has no non-null value
```

```
"""
df_andhra.drop('pm2_5', axis = 1, inplace = True)

df_andhra['rspm'].fillna(df_andhra['rspm'].mean(), inplace = True)
df_andhra['spm'].fillna(df_andhra['spm'].mean(), inplace = True)

sns.relplot(y="no2", x="so2",
data=df_andhra);

"""
Changing the format of the date column in df_andhra to datetime format
for the ease of calculation and further process
"""
df_andhra['date'] = pd.to_datetime(df_andhra['date'], format = "%m/%d/%Y")

"""
Setting date as the index of df_andhra dataframe
"""
df_andhra.set_index('date', inplace = True)

df_andhra.drop('state',axis = 1, inplace = True)

sns.pairplot(df_andhra)

"""
Encoding the data into numerical format as :
1 - ML models need data to be in numercial format
"""
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()

for col in df_andhra:

# Compare if the dtype is object

if df_andhra[col].dtype=='object':

# Use LabelEncoder to do the numeric transformation

df_andhra[col]=le.fit_transform(df_andhra[col])


"""

Creating seperate columns for encoded data

"""

from sklearn.preprocessing import OneHotEncoder


enc = OneHotEncoder(handle_unknown='ignore')


enc_df = pd.DataFrame(enc.fit_transform(df_andhra[['location']]).toarray())


x = df_andhra.index

enc_df['date'] = x


enc_df.set_index('date', inplace = True)

enc_df1 = pd.DataFrame(enc.fit_transform(df_andhra[['type']]).toarray())

enc_df1


y = df_andhra.index

enc_df1['date'] = y


enc_df1.set_index('date', inplace = True)

enc_df1.rename(columns = {0 : 'a', 1 : 'b', 2: 'c'}, inplace = True)

for i in range(0, 24):
```

```
df_andhra[i] = enc_df[i]

df_andhra['a'] = enc_df1['a']

df_andhra['b'] = enc_df1['b']

df_andhra['c'] = enc_df1['c']


df_andhra['no2'].isna().sum()

df_andhra['no2'].fillna(df_andhra['no2'].mean(), inplace = True)


y = df_andhra.iloc[:, 2:3].values

df_andhra.reset_index()

y.reshape(1,-1)

df_andhra.drop('so2', axis = 1, inplace = True)

df_andhra.drop('location', axis = 1, inplace = True)


X = df_andhra.values

y.shape


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣

,→random_state = 23)


from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()

X_train = sc_X.fit_transform(X_train)

X_test = sc_X.transform(X_test)


sc_y = StandardScaler()

y_train = sc_y.fit_transform(y_train)
```

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 1000, random_state = 0)

regressor.fit(X, y)

y_pred = regressor.predict(X_test)

predictor = regressor.predict(X_train)


from sklearn.metrics import r2_score

r2_score(y_test, y_pred)
```

Observations –

- It was observed that, Delhi had the highest concentration of rspm and spm values.
- The state with the highest concentration of SO2 was Uttarakhand
- The state with the highest concentration of NO2 was West-Bengal
- We can see that no2 and so2 have a somewhat similar pattern with other features. It can be said that spm and rspm share somewhat linear relationship, rest all features are not entirely related.